S-BPM – A NEW IMPETUS IN BUSINESS PROCESS MANAGEMENT

Albert Fleischmann, Interaktiv Unternehmensberatung
Werner Schmidt, Technische Hochschule Ingolstadt
Welcome to the Institute of Innovative Process Management (I2PM) e.V.

You landed on the community platform of I2PM, which is a registered non-profit association.

This platform provides content and interaction features open to:

- all internet users
- registered members of the I2PM community (if interested to join sign up in the upper right corner)
- registered members the I2PM association

with access rights depending on their specific status.

The I2PM association was established to promote innovative scientific discoveries and solutions in the field of process management and to test them through academic work over the long term. The objective is to transfer know-how from theory to practice.

Specifically, that means:

- We manage and foster scientific work.
- We organize events to share knowledge and experience.
- We publish findings and results.
- We support research and development projects.
- We cooperate with other academic organizations, associations, institutions and enterprises.
Outline

• Challenges in BPM
• What is S-BPM?
  – Constituents
  – Overall Approach
  – Impacts
• S-BPM Notation
• S-BPM Life Cycle
  – Analyzing & Modeling
  – Validating & Optimizing
  – Embodying
  – Running & Monitoring
• S-BPM Benefits revisited
• Future Perspectives
Challenges in BPM

From traditional to contemporary BPM

**Traditional BPM**

**Characteristics**
- Based on Taylor/Ford
- Activity-driven
- Control flow-oriented
- Highly structured, well-defined processes
- Languages/notations: e.g., BPMN, BPEL

**Issues**
- Model-reality divide
- Lost innovation
- Information pass-on threshold
- Lack of information fusion

**Contemporary BPM**

**Characteristics**
- Based on PostFordism
- Stakeholder-driven
- Communication-oriented
- Less-structured flexible processes
- Languages/notations: ?

**Challenges**
- Semantic integration
- Organizational integration
- BPM Life Cycle responsiveness

What can Subject-oriented BPM (S-BPM) contribute?
Introduction to S-BPM
What is it?

• **S-BPM** means **Subject-oriented Business Process Management**
• Methodology for all BPM activities (from analysis & modeling to execution & monitoring)
• Basis regarding the user (mainly modeler):
  – Standard semantics of natural language with subject, predicate, and object
  – Easy-to-use notation with only a few symbols
• Basis regarding information technology:
  – Process algebra with a clear formal semantic
  – allowing automated code generation and models being executable on the fly
• **Origin**: Albert Fleischmann
• **Tool environment**
  – **Metasonic Suite** (by Metasonic GmbH)
  – **InFlow** (by StrICT Solutions GmbH)
  – **Open S-BPM Initiative** as a platform for development of other/additional tools
Change the Mindset: From Sequence to Parallelism

**Reality:**
World is a concurrent system. Due to division of labour actors work parallel and need to interact in order to accomplish tasks.

**Problem:**
Today world is perceived as a sequence of actions.

**Change:**
Make people see the world as it is – parallel.
Change the Mindset: From Sequence to Parallelism

Reality:
World is a concurrent system. Due to division of labour actors work parallel and need to interact in order to accomplish tasks.

Problem:
Today world is perceived as a sequence of actions.

Change:
Make people see the world as it is – parallel.
Introduction to S-BPM
Notation

Subject Interaction Diagram (SID)
Introduction to S-BPM Notation

Subject Behavior Diagram (SBD)
Introduction to S-BPM

Notation

Process Network Diagram (PND)
S-BPM Life Cycle
Activity Bundles & Roles

Governors
Actors
Facilitators
Experts
S-BPM Life Cycle
Example for Tool Support

Architecture of Metasonic Suite

Modeling
Validation
Embedding into organization
Execution

Model Manager
- Models
- Business Objects

User Manager

Instance Manager

SysInfo

LDAP (AD)
S-BPM Life Cycle
Analyzing & Modeling

Modeling Steps

Starting point: Process description in natural language

An employee fills in a business trip request form. He/She puts in start date, end date and reason for his/her trip. The responsible manager checks the application and informs the employee about his/her decision; the business trip request might be rejected or get approved. In case of approval the trip data is send to the travel agent which processes it.

Steps

• Identification and description of the subjects involved in the process

• Identification and description of interactions the subjects are part of

• Specification of the messages they send or receive and work on (parameters, business objects)

• Detailing the behavior of each subject

Subject Interaction Diagram

Subject Behavior Diagram
UI Types for Elicitation and Representation

Brown Paper

T-BPM

GUI-based
S-BPM Life Cycle
Analyzing & Modeling

UI Types for Elicitation and Representation

Build Book (S-BPM-based)
UI Types for Elicitation and Representation

Comprehand (S-BPM-based)  
Metasonic Touch (S-BPM-based)
UI Types for Elicitation and Representation – Benefits of S-BPM

<table>
<thead>
<tr>
<th>BPM Activity</th>
<th>Modeling</th>
<th>Sustainability of Documentation</th>
<th>Implementation</th>
<th>Execution (IT-based)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Analysis, Acquisition, Elicitation</td>
<td>Comprehend (S-BPM-based)</td>
<td>Automated workflow generation</td>
<td>Partially automated workflow generation</td>
<td>Programming (e.g., in BPEL)</td>
</tr>
<tr>
<td></td>
<td>Rural Comprehend (S-BPM-based)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Buildbook game (S-BPM-based)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>T-BPM (BPMN-based)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Brown paper</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
S-BPM Life Cycle
Validating & Optimizing

Check of Effectiveness and Efficiency (Interaction and Behavior)

Traditional Walkthrough

S-BPM- and computer-based role play
S-BPM Life Cycle
Validating & Optimizing

Possible Level Concept

Validation Level

III
Process view
(overall process)

II
Interaction view
(sub-processes)

I
Subject view
(individual behavior)

Underlying S-BPM models (diagrams)

Process Network Diagram

Subject Interaction Diagram

Subject Behavior Diagram

Involved S-BPM actors

Approval
Actor(s)
Facilitator(s)
Governor(s)
Customer(s)

© i2PM.net
S-BPM Life Cycle
Embodying

Organization – Staffing Subjects

Org Chart

Routing of BT Request ‘Schulz’

Subject Representation Table (BT Request ‘Schulz’)

<table>
<thead>
<tr>
<th>Subject</th>
<th>Subject behavior</th>
<th>Subject carrier</th>
<th>Organizational unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Employee</td>
<td>is transferred to</td>
<td>Mr. Schulz</td>
<td>Sales</td>
</tr>
<tr>
<td>Manager</td>
<td>is transferred to</td>
<td>Mr. Schmid</td>
<td>Sales</td>
</tr>
<tr>
<td>Travel office</td>
<td>is transferred to</td>
<td>Mr. Way</td>
<td>HR</td>
</tr>
</tbody>
</table>
S-BPM Life Cycle
Embodying
Organization – Staffing Subjects

Routing of BT Request ‘Black’

Organizational Context of BT Request

<table>
<thead>
<tr>
<th>Employee start subject</th>
<th>Manager in organizational context</th>
<th>Travel office in organizational context</th>
</tr>
</thead>
<tbody>
<tr>
<td>Miller</td>
<td>Miller</td>
<td>Way</td>
</tr>
<tr>
<td>Schmid</td>
<td>Miller</td>
<td>Way</td>
</tr>
<tr>
<td>Meier</td>
<td>Miller</td>
<td>Way</td>
</tr>
<tr>
<td>Schulz</td>
<td>Schmid</td>
<td>Way</td>
</tr>
<tr>
<td>Huber</td>
<td>Schmid</td>
<td>Way</td>
</tr>
<tr>
<td>Black</td>
<td>Meier</td>
<td>Way</td>
</tr>
<tr>
<td>Way</td>
<td>Schmid</td>
<td>Way</td>
</tr>
</tbody>
</table>
S-BPM Life Cycle
Embodying

Organization – Staffing Subjects

Deputyship

Routing depending on business object values

<table>
<thead>
<tr>
<th>Employee (start subject)</th>
<th>Manager in organizational context</th>
<th>Travel office in organizational context</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Schmid</td>
<td>If requested destination country = Germany, then Way, otherwise Langway</td>
</tr>
<tr>
<td></td>
<td>Schmid</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Schmid</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Schmid</td>
<td></td>
</tr>
</tbody>
</table>
**S-BPM Life Cycle**

**Embodying**

**Organization – Staffing Subjects**

**Grouping of Subject Representatives**

- Management board
  - Miller
- Development/Meier
- Sales
  - Schmid
- Black
  - Lange
  - Shortway
- Huber

**Organizational Context of BT Request**

<table>
<thead>
<tr>
<th>Employee start subject</th>
<th>Manager in organizational context</th>
<th>Travel office in organizational context</th>
</tr>
</thead>
<tbody>
<tr>
<td>Miller</td>
<td>Miller</td>
<td>Way</td>
</tr>
<tr>
<td>Schmid</td>
<td>Miller</td>
<td>Travel Office</td>
</tr>
<tr>
<td>Meier</td>
<td>Miller</td>
<td>Travel Office</td>
</tr>
<tr>
<td>Schultz</td>
<td>Schmid</td>
<td>Travel Office</td>
</tr>
<tr>
<td>Huber</td>
<td>Schmid</td>
<td>Travel Office</td>
</tr>
<tr>
<td>Rank</td>
<td>Meier</td>
<td>Travel Office</td>
</tr>
<tr>
<td>Way</td>
<td>Schmid</td>
<td>Travel Office</td>
</tr>
</tbody>
</table>
S-BPM Life Cycle
Embodying

Organization – Example: Subject Assignment in Metasonic Suite
S-BPM Life Cycle
Embodying

Organization – Example: Deputyship in Metasonic Flow
### S-BPM Life Cycle

**Embodying**

### IT - Implementing Workflow Application

<table>
<thead>
<tr>
<th>Workflow characteristic</th>
<th>Representation in S-BPM</th>
<th>Implementation in IT</th>
</tr>
</thead>
</table>
| Actors (roles/persons)   | * Subjects/subject groups  
                          * Subject carrier (result of the organizational implementation) | Implementation of subjects and subject carriers via the user management of systems (e.g., via LDAP, role and rights concepts, etc.) |
| Actors (IT system)       |                         | Implementation of the action behavior of non-human-subjects (integration of existing and new applications, e.g., as Web Services) |
| Activities controlled by business logic and rules | * Subject behavior (internal strictly sequential → orchestration)  
                          * Communication structure (subject interactions → choreography) | Implementation of the action and communication behavior of subjects as steps in Workflow Engine / Business Rule Engine with the integration of existing and new applications (e.g., as Web services) |
| Communication of the actors | Communication structure (subject interactions), including synchronization of message exchanges using the input pool | Implementation of the communication behavior of the subjects, e.g., by e-mail, etc. |
| Objects (inputs and outputs incl. data structures) | Business objects | Implementation of business objects and their manipulation in the action behavior of the subjects (by user interaction (screen forms) or automated application functionality (e.g., program-triggered database transactions) |
S-BPM Life Cycle

Embodying

IT - Integrating Applications

States in subject behavior can be used for

• calling external applications (e.g., SAP)
• including code by refinements
Summary: S-BPM Contribution to contemporary BPM

Organizational Integration
- Intensive stakeholder participation (inclusion):
  - All people speaking natural language (even methodology/technology illiterates)
  - Actors can easily model, test and improve their processes
  - People in all roles can drive organizational development

Semantic Integration
- Mutual understanding and stakeholder empowerment
  - Simple, ‘social’ notation (only a few symbols, natural language-like)
  - Executable model → single model (business & IT) → seamless roundtrip engineering

Life Cycle Responsiveness
- Linear and non-linear, flexible order of activities
- Change initiative from all roles
- Fast validation/implementation as workflow
- Direct feedback during execution can trigger short-term changes/troubleshooting

Benefits
- Improved model quality, reduced complexity and coherent overall specification
- Minimal model-reality divide (‘what you model is what you get’) → stakeholder acceptance
- Agile organizational development (short cycles, short time to execution, seamless roundtrip)
- Stakeholder-driven organizational learning
- Self organization (maximal individual freedom and assured coherence)
- Bottom-up and top-down and mixed process design (centrally/decentrally initiated)
- ...

www.i2PM.net
© I2PM

HSE 2014
S-BPM - New Impetus in Business Process Management
Future Perspectives

• Agility requires Simplicity
  • Modeling in a simple way with an available Tool
  • From the Model to Code: New Programming Languages and frameworks
• Simplicity with ASM Networks?
• Summary
SID of Automatic Teller Machine (ATM)

Customer
- Give Pin
- Pin correct
- Pin incorrect Credit card is ke
- Pin incorrect try aga
- Credit Card
- confirmation of post state
- Balance
- next action
- Money
- Amount to hig

ATM
- Credit Carc
- PIN
- Give Balance
- Post statemer
- Withdraw
- Finishec
- Money remove
- interrupt-transacti

Database manager
- Check credit car
- money withdrawl check li
- money withdraw
- Lock credit Carr
- Balancedat:rs
- Interrupt-Transacti

Post service
- Send Balance lett:

Balance statement

www.i2PM.net © i2PM
HSE 2014
S-BPM - New Impetus in Business Process Management
ATM:
Insert credit card and check its validity
Select and execute action
There are not enough bills in the ATM: Additional internal action has to be inserted.
Drawing is cumbersome!!!!
<table>
<thead>
<tr>
<th>Attribute</th>
<th>Customer</th>
<th>ATM</th>
<th>Database Manager</th>
<th>Postal Service</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subject name</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Messages</td>
<td>Credit card</td>
<td>-&gt;</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>PIN</td>
<td>-&gt;</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Give Balance</td>
<td>-&gt;</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Post statement</td>
<td>-&gt;</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Withdrawal</td>
<td>-&gt;</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Finished</td>
<td>-&gt;</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Money remove</td>
<td>-&gt;</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Interrupt Transaction</td>
<td>-&gt;</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Check credit card</td>
<td>-&gt;</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>monex withdrawal check limit</td>
<td>-&gt;</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>money withdrawal</td>
<td>-&gt;</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Lock credit card</td>
<td>-&gt;</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Balancedata</td>
<td>-&gt;</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Interrupt transaction</td>
<td>-&gt;</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>&lt;- Accept credit card</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>&lt;- Accept credit card</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>&lt;- Accept credit card</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>&lt;- Accept credit card</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>&lt;- Accept credit card</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>&lt;- Accept credit card</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>&lt;- Accept credit card</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>&lt;- Accept credit card</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>&lt;- Accept credit card</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>&lt;- Give PIN</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>&lt;- PIN correct</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>&lt;- incorrect Credit card is kept</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>&lt;- PIN incorrect try again</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>&lt;- Credit card</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>&lt;-formation of post statement</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>&lt;-Balancedata</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>&lt;-next action</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>&lt;-money withdrawal</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>&lt;-Amount to high</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Send balance letter</td>
<td></td>
<td>-&gt; Balance statement</td>
<td></td>
</tr>
</tbody>
</table>
### ATM Behaviour in Excel (1)

<table>
<thead>
<tr>
<th>Attribute Type</th>
<th>E</th>
<th>S</th>
<th>E</th>
<th>S</th>
<th>E</th>
<th>S</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stages</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>get credit card</td>
<td>PIN</td>
<td>wait for PIN</td>
<td>check PIN</td>
<td>credit card</td>
<td>Credit card ok</td>
<td>Action</td>
<td></td>
</tr>
<tr>
<td>(Customer)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Credit card</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Customer)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>give PIN</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Customer)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PIN</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Customer)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Check credit card</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Data base Manager)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>accept credit card</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Data base Manager)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PIN correct</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Customer)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PIN not correct</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Data base Manager)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>credit card locked</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(&lt;)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(&lt;)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## ATM Behaviour in Excel (2)

<table>
<thead>
<tr>
<th>E</th>
<th>S</th>
<th>E</th>
<th>E</th>
<th>E</th>
<th>S</th>
<th>S</th>
<th>S</th>
<th>I</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Action</strong></td>
<td><strong>Balance</strong></td>
<td><strong>Postal service</strong></td>
<td><strong>Balance</strong></td>
<td><strong>Postal Balance</strong></td>
<td><strong>Handover Balance</strong></td>
<td><strong>postal balance</strong></td>
<td><strong>next</strong></td>
<td><strong>Give credit card back</strong></td>
<td><strong>end money</strong></td>
</tr>
<tr>
<td>(Customer) give balance</td>
<td>-&gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Customer) post statement</td>
<td>-&gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Customer) withdrawal</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Database Manager) balance data</td>
<td>-&gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>(Database Manager) money withdrawal check limit</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### ATM Behaviour in Excel (3)

<p>| | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>E</td>
<td>S</td>
<td>E</td>
<td>S</td>
<td>S</td>
<td>I</td>
<td></td>
<td></td>
</tr>
<tr>
<td>money</td>
<td>limit</td>
<td>Give money</td>
<td>Money removed</td>
<td>PIN again</td>
<td>credit card locked</td>
<td>end credit card is locked</td>
<td>→</td>
<td></td>
</tr>
<tr>
<td>→</td>
<td>(Database Manager) money withdrawal check limit</td>
<td>→</td>
<td>(Database Manager) withdrawal allowed</td>
<td>→</td>
<td>(Customer) money</td>
<td>→</td>
<td></td>
<td></td>
</tr>
<tr>
<td>→</td>
<td>(Customer) money removed</td>
<td>→</td>
<td>(Customer) PIN incorrect Credit card is kept</td>
<td>→</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Customer) PIN incorrect try again</td>
<td>→</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Attribute</td>
<td>Customer</td>
<td>ATM</td>
<td>Database Manager</td>
<td>Postal Service</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---------------------------</td>
<td>----------------</td>
<td>-----------------</td>
<td>------------------</td>
<td>----------------</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Subject</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Messages</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Credit card</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PIN</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Give Balance</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Post statement</td>
<td>&lt;-</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Withdrawal</td>
<td>&lt;-</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Finished</td>
<td>&lt;-</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Money remove</td>
<td>&lt;-</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Interrupt Transaction</td>
<td>&lt;-</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Check credit card</td>
<td>&lt;-</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Money withdrawal check lin</td>
<td>&lt;-</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Money withdrawal</td>
<td>&lt;-</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lock credit card</td>
<td>&lt;-</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Balanced data</td>
<td>&lt;-</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Interrupt transaction</td>
<td>&lt;-</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>&lt;- Accept credit card</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>&lt;- Accept credit card</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>&lt;- Accept credit card</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>&lt;- Accept credit card</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>&lt;- Accept credit card</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>&lt;- Accept credit card</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>&lt;- Accept credit card</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>&lt;- Accept credit card</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>&lt;- Accept credit card</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>&lt;- Accept credit card</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>&lt;- Accept credit card</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>&lt;- Accept credit card</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>&lt;- Accept credit card</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>&lt;- Give PIN</td>
<td>&lt;-</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>&lt;- PIN correct</td>
<td>&lt;-</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>&lt;- Incorrect Credit card is kept</td>
<td>&lt;-</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>&lt;- PIN incorrect try again</td>
<td>&lt;-</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>&lt;- Credit card</td>
<td>&lt;-</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>&lt;- Formation of post statement</td>
<td>&lt;-</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>&lt;- Balanced data</td>
<td>&lt;-</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>&lt;- Next action</td>
<td>&lt;-</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>&lt;- Money withdrawal</td>
<td>&lt;-</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>&lt;- Amount to high</td>
<td>&lt;-</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Send balance letter</td>
<td>&lt;-</td>
<td></td>
<td></td>
<td>Balance statement</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>&lt;- Not enough bills available</td>
<td>&lt;-</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Change ATM Behaviour: Enough Bills available

<table>
<thead>
<tr>
<th>S</th>
<th>E</th>
<th>S</th>
<th>S</th>
<th>L</th>
</tr>
</thead>
<tbody>
<tr>
<td>Give money</td>
<td>Money removed</td>
<td>PIN again</td>
<td>Credit card locked</td>
<td>end credit card is locked</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Customer)money</td>
<td>-&gt;</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>&lt;-</td>
<td></td>
<td>In enough bills available</td>
<td>Not enough bills available</td>
<td>-&gt;</td>
</tr>
<tr>
<td>(Customer)money removed</td>
<td>-&gt;</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>(Customer)PIN incorrect Credit card is kept</td>
</tr>
<tr>
<td>(Customer)PIN incorrect try again</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S</td>
<td>E</td>
<td>S</td>
<td>S</td>
<td>I</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>Give money</td>
<td>Money removed</td>
<td>PIN again</td>
<td>credit card locked</td>
<td>end credit card is locked</td>
</tr>
<tr>
<td>(Customer)Credit card locked</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Customer)money removed</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Customer)money</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Customer)money removed</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Customer)PIN incorrect try again</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
ATM Behavior in CSV Format

| Attribute | Type | E | S | E | S | E | S | E | S | I | S | E | S | S | S | I | S | E | S | S | I |
| Status    |      |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| get credit card | PIN | wait for PIN | check PIN | credit card | credit card ok | Action | Balance | ……… | (Customer) Credit card | -> | (Customer) give balance | -> | ……… | (Customer) Credit card | -> | (Customer) give PIN | -> | (Customer) post statement | -> | ……… |
| (Customer) PIN | -> | (Customer) withdrawal | ……… | (Data base Manager) Check credit card | -> | (Database Manager) balance data | -> | ……… | (Data base Manager) accept credit card | -> | (Database Manager) balance data | -> | ……… | (Customer) PIN correct | -> | (Customer) balance | -> | (Customer) money | -> | ……… |
| ……… | <- | (Customer) next action | ……… | (Database Manager) balance data | -> | ……… | (Database Manager) balance data | -> | ……… | (postal service) send balance letter | -> | ……… | (Customer) money removed | <- | ……… | (Customer) PIN not correct | ……… | -> | ……… | (Data base Manager) credit card locked | -> | ……… | (Customer) Pin incorrect Credit card is kept | -> | <- | (Customer) PIN incorrect try again | <- | ……… |
Code structure subject

Subject

Code patterns
Code for behavior
Subject specific

Applications and lokal data
Subject specific

Input pool

Deposit
Subject B

Deposit
Subject N

Use

Subject C

Deposit
Subject K

Deposit
Subject K

Remove
Implementation send

```
case state name  => 
    //Deposit message in the
    //input pool of receiving subject
    S1.deposit (A)
    // Set Current state to the
    // succeeding state
    CurrentState = Next state
```
case State name =>
{
  // Check whether message A from subject S1 is in the input pool
  if (InputPool.lookup((S1)A)) {
    // If it is in the input pool pick it up and set current state to succeeding state
    InputPool.pickup ((S1)A)
    CurrentState = Next state 1
  }
  else if
  // Check whether message B from subject S2 is in the input pool
  (InputPool.lookup((S2)B)) {
    InputPool.pickup ((S2)B)
    CurrentState = Next state 2
  }
case State name =>

{ // execute internal function
    result = Internal function
    if (result == Result 1) {
        // If function result is Result 1 set current state to corresponding succeeding state
        CurrentState = Next state 1
    } else if (result == Result 2) {
        // If function result is Result 1 set current state to corresponding succeeding state
        CurrentState = Next state 2
    }
}
Simple Input Pool Implementation.

class InputPool (name : String) {
  var store : List[message] = Nil

  // send messages
  def deposit (sendMessage : message) {store = sendMessage::store}

  def lookup (lookMessage : message) : Boolean = lookupMessage (store, lookMessage)

  //Receive messages
  //Check whether message is in the input pool
  def lookupMessage (Mails : List[message], lookMessage : message) : Boolean = {
    Mails match {
      case firstmessage::xs => { if (firstmessage == lookMessage) true else lookupMessage(xs, lookMessage)}
      case Nil => false
    }
  }

  //Remove message from the input Pool
  def pickup (lookMessage : message) = {store = remove (store, lookMessage)}

  def remove (Mails : List[message], lookMessage : message) : List[message] = {
    Mails filter (x => x != lookMessage)
  }
}
Coffee example

Programmer  Coffee machine

Make coffee  Coffee finished
Behavior Programmer

Programmer:

Start coffee machine

Check milk and sugar

Goto the store
Buy cream and sugar
Go home
Coffee finished
Drink coffee

(Coffee machine) Make coffee

Not available
Store entered
Bought cream and sugar
At home

(Coffee machine) Coffee finished

Bought cream and sugar

Available
Not available

HSE 2014
def executeProgrammerBehavior {
    println("")
    println("Execute: " + name + " State is: " + CurrentState)
    CurrentState match {
        case Check_milk_and_sugar => {
            println("Current State is: " + CurrentState)
            FunctionResult = InternalFunctions.Check_milk_and_sugarFunction
            println("Function Result:" + FunctionResult)
            CurrentState = Go_to_the_store
        }
        case StartCoffeeMachine => {
            println("Current State is: " + CurrentState)
            CoffeeMachinesInputPool.deposit(SendMakeCoffeeToCoffeeMachine)
            CurrentState = Check_milk_and_sugar
        }
        case Go_to_the_store => {
            println("Current State is: " + CurrentState)
            FunctionResult = InternalFunctions.Go_to_the_storeFunction
            println("Function Result:" + FunctionResult)
            CurrentState = Buy_cream_and_sugar
        }
        case Buy_cream_and_sugar => {
            println("Current State is: " + CurrentState)
            FunctionResult = InternalFunctions.Buy_cream_and_sugarFunction
            println("Function Result:" + FunctionResult)
            CurrentState = Go_home
        }
        case Go_home => {
            println("Current State is: " + CurrentState)
            FunctionResult = InternalFunctions.Go_homeFunction
            println("Function Result:" + FunctionResult)
            CurrentState = Coffee_finished
        }
        case Coffee_finished => {
            println("Current State is: " + CurrentState)
            println("Jetzt kommt empfangen")
            if (ProgrammersInputPool.lookup(ReceiveCoffeeFinished)) {
                ProgrammersInputPool.pickup(ReceiveCoffeeFinished)
                CurrentState = Drink_coffee
            } else {
                println("Message Coffee finished not in InputPool")
            }
        }
        case Drink_coffee => {
            println("Current State is: " + CurrentState)
            FunctionResult = InternalFunctions.Drink_coffeeFunction
            println("Function Result:" + FunctionResult)
            CurrentState = EndState
        }
        case _ => println("Invalid state")
    }
}
class ProgrammerInternalFunctions
// This is the interface to existing applications used by subject programmer
{
    def Check_milk_and_sugarFunction : String = "Result of function Check_milk_and_sugar is: not available"
    def Go_to_the_storeFunction : String = "Result of function Go_to_the_store is: Store entered"
    def Buy_cream_and_sugarFunction : String = "Result of function Buy_cream_and_sugar is: Bought cream and sugar"
    def Go_homeFunction : String = "Result of function Go_home is: at home"
    def Drink_coffeeFunction : String = "Result of function Drink_coffee is: coffee is empty"
Code structure subject

System node X

Scheduler: Main program

Invoke

Return

Invoke

Return

Invoke

Return

Class for subject A with behavior method

Class for subject B with behavior method

Class for subject C with behavior method
object Scheduler {

// Context 1
val Programmers1InternalFunctions = new ProgrammerInternalFunctions
val Programmers1InputPool = new InputPool ("Input Pool Programmer")

val CoffeeMachines1InputPool = new InputPool ("Input Pool Coffee machine")
val Coffeemachines1InternalFunctions = new CoffeeMachineInternalFunctions

val CoffeeMachine1 = new CoffeeMachineBehavior (Programmers1InputPool, CoffeeMachines1InputPool, Coffeemachines1InternalFunctions, "coffeeMachine1")
val Programmer1 = new ProgrammerBehavior (Programmers1InputPool, CoffeeMachines1InputPool, Programmers1InternalFunctions, "Programmer 1")

// Dispatcher for subject behavior. Can be also a while loop.
// If subjects are implemented as actors a dispatcher is not necessary
def dispatch {
    Programmer1.executeProgrammerBehavior
    Programmer1.executeProgrammerBehavior
    CoffeeMachine1.executeCoffeeMachineBehavior
    CoffeeMachine1.executeCoffeeMachineBehavior
    CoffeeMachine1.executeCoffeeMachineBehavior
    CoffeeMachine1.executeCoffeeMachineBehavior
    CoffeeMachine1.executeCoffeeMachineBehavior
    CoffeeMachine1.executeCoffeeMachineBehavior
    CoffeeMachine1.executeCoffeeMachineBehavior
    CoffeeMachine1.executeCoffeeMachineBehavior
    CoffeeMachine1.executeCoffeeMachineBehavior
    CoffeeMachine1.executeCoffeeMachineBehavior
    CoffeeMachine1.executeCoffeeMachineBehavior
    CoffeeMachine1.executeCoffeeMachineBehavior
    CoffeeMachine1.executeCoffeeMachineBehavior
    CoffeeMachine1.executeCoffeeMachineBehavior
    CoffeeMachine1.executeCoffeeMachineBehavior
    CoffeeMachine1.executeCoffeeMachineBehavior
    CoffeeMachine1.executeCoffeeMachineBehavior
    CoffeeMachine1.executeCoffeeMachineBehavior
    CoffeeMachine1.executeCoffeeMachineBehavior
    CoffeeMachine1.executeCoffeeMachineBehavior
    CoffeeMachine1.executeCoffeeMachineBehavior
    CoffeeMachine1.executeCoffeeMachineBehavior
    CoffeeMachine1.executeCoffeeMachineBehavior
    CoffeeMachine1.executeCoffeeMachineBehavior
    CoffeeMachine1.executeCoffeeMachineBehavior
    CoffeeMachine1.executeCoffeeMachineBehavior
    CoffeeMachine1.executeCoffeeMachineBehavior
    CoffeeMachine1.executeCoffeeMachineBehavior
    CoffeeMachine1.executeCoffeeMachineBehavior
    CoffeeMachine1.executeCoffeeMachineBehavior
    CoffeeMachine1.executeCoffeeMachineBehavior
    CoffeeMachine1.executeCoffeeMachineBehavior
    CoffeeMachine1.executeCoffeeMachineBehavior
    CoffeeMachine1.executeCoffeeMachineBehavior
    CoffeeMachine1.executeCoffeeMachineBehavior
    CoffeeMachine1.executeCoffeeMachineBehavior
    CoffeeMachine1.executeCoffeeMachineBehavior
    CoffeeMachine1.executeCoffeeMachineBehavior
}
}
Code structure subject
Actors in Akka

Outside world Sends messages to Actor reference

Actor (your code) Proxies for

Messages contains

Mailbox

Thread

Dispatcher

Puts Mailbox on

passes message to

Delegates queueing
ABSTRACT STATE MACHINE NETWORKS: A MORE ABSTRACT VIEW ON SYSTEMS?
ASM Net Transition

Mode/Phase End/start of a phase
Condition (Sachverhalt)
Action

i(1) → EntryCond(1) → ExitCond(1) → j(1)

i(n) → EntryCond(n) → ExitCond(m) → j(m)

...
Open Issues

• Is it understandable for all stakeholders?

• Is it necessary to develop a new graphical representation?

• ........
Summary

• Use tools which are commonly available for modeling

• Create code which can be executed directly without a central workflow engine.

• Investigation of ASM Networks
More Details on S-BPM

Books

Fleischmann, A., Schmidt, W., Stary, C., Obermeier, S., Börger, E.  
Subject-oriented Business Process Management  
Springer, Berlin Heidelberg 2012,  
available for free under open access license at:  
http://link.springer.com/book/10.1007/978-3-642-32392-8/page/1

Fleischmann, A., Schmidt, W., Stary, C., Obermeier, S., Börger, E.  
Subjektorientiertes Prozessmanagement – Mitarbeiter einbinden, Motivation und Prozessakzeptanz steigern  
Hanser, München 2011

Fleischmann, A., Raß, S., Singer, R.  
S-BPM Illustrated  
Springer, Berlin Heidelberg, in press, appears in May 2013,  
available for free under open access license at:  
http://link.springer.com/book/10.1007/978-3-642-36904-9/page/1
More Details on S-BPM Books

Fleischmann, A., Schmidt, W., Stary, C., (Eds.)
S-BPM in the Wild – Value Creating Practice in the Field
Springer, Berlin Heidelberg 2015,
More Details on S-BPM
S-BPM ONE Conference Series

Karlsruhe 2009
Karlsruhe 2010
Ingolstadt 2011
Vienna 2012
Deggendorf 2013
Eichstätt 2014

Call for Papers available

WWW.S-BPM-ONE.ORG
More Details on S-BPM
Other Scientific Activities

Workshop on Cross-organizational and cross-company BPM (Xoc-BPM) in conjunction with IEEE Conference on Business Informatics (CBI)

Workshop on Agent Systems in Business Process Management (AS-BPM) in conjunction with IEEE/WIC/ACM International Conference on Intelligent Agent Technology (WI-IAT)

Vienna 2013
Geneva 2014
Lisbon 2015
Atlanta 2013
Warsaw 2014
Singapore 2015

Calls for Papers coming soon